Abstract - In the absence of absolute localization tools such as GPS, a robot can still successfully navigate by conducting Simultaneous Localization and Mapping (SLAM). All SLAM algorithms to date can only be applied in one environment at a time. In this paper we propose to extend SLAM to multi-environments. In SmartSLAM, the robot first classifies its entourage using environment recognition code and then performs SLAM using landmarks that are appropriate for its surrounding milieu. One thousand images of various indoor and outdoor environments were collected and used as training data for a three-layered feedforward backpropagation neural network. This neural network was then tested on two sets of query images of indoor environments and another two sets of outdoor environments, yielding 83% and 95% correct classification rates for the indoor images and 80% and 79% success rates for the outdoor images.

Keywords: SLAM, multi-environments, context priming, computer vision, machine learning.

1 Introduction

To insure complete autonomy, a robot must be able to map unknown environments and localize itself with respect to this map. This problem was formally coined ‘Simultaneous Localization and Mapping’ (SLAM) [1] and has been an area of active research since the beginning of the past decade.

In order to perform SLAM the robot uses features that it can recognize in its entourage to construct a map. In all SLAM algorithms developed so far, the shape and properties of these features are clearly defined to the robot before it initiates SLAM. The major disadvantage with such algorithms is that the robot is confined to navigate in one class of environments; one that contains a sufficient amount of these pre-defined landmarks. In other words, if the robot were to inadvertently navigate out of this setting and into another that includes none of the expected features, SLAM would fail.

In this paper SLAM is developed to a higher level of autonomy by having the robot select features that are suitable for the setting it is located in. This goal is achieved by incorporating environment recognition code into the SLAM algorithm. Rather than attempting to perform environment recognition by the conventional bottom-up approach, which is too time-consuming to be applied in real-time robot navigation, a top-down, holistic approach is adopted. Images are classified according to their low level spatial structure.

At a preprocessing stage, a large number of images of various classes (e.g. indoor office, indoor corridor, indoor hall, outdoor street) are collected. A steerable pyramid (SP) filter is applied to each of these images in order to extract its low level feature vector. Principal Component Analysis (PCA) is then performed on all these vectors to reduce their dimensionality. The feature vectors, along with the class of each image will then be used to train a three-layered feedforward Neural Network (NN). Once the NN is sufficiently trained the robot can initiate navigation. Equipped with a vision system, the robot acquires images from its surrounding environment and classifies one hundred of them using the trained NN. The class with the highest votes is picked as the recognized environment. The scope of this paper is limited to two environments: indoor and outdoor. Once environment classification is achieved, the robot is assigned features that agree with the recognized setting. Typical features could include trees for outdoor parks and corners for indoor settings. During SLAM, the robot periodically scans its surrounding to insure that it is still located within the same environment. If not, it attempts to recognize its new setting and picks a feature type that is appropriate for this new setting.

This paper is organized as follows. In Section 2 the SLAM problem is briefly described. An historical overview is presented that covers the type of features that have been used to perform SLAM. Section 3 formalizes the framework for environment recognition: Extracting the feature vectors of images, training the neural network,
and classifying query images. Section 4 explicates the SmartSLAM algorithm. Section 5 presents the results of the tests that were conducted on SmartSLAM. In section 6 relevant conclusions are drawn and the direction of future research is laid out.

2 SLAM

When a robot navigates in an unknown environment, its proprioceptive sensors are used to estimate its dynamics and its exteroceptive sensors collect relative position estimates of landmark locations. These estimates are correlated because of the common error in estimating position estimates of landmark locations. These estimates were the beacons, or landmarks, that were used were chosen’s ‘meet-points’, which correspond to intersections, corners, or dead ends in corridors. From 1999 to date, there have been a myriad of papers on SLAM and different approaches of solving it. Listing all these papers would be futile. Rather, only the ones that are the most relevant to this paper will be mentioned. In 2002, DiSana et al. [6] showed that it is possible to remove a great deal of landmarks from the SLAM map without making the map building process statistically inconsistent. The landmarks that the authors used were stationary point targets, such as foreground points, corners and edges. A laser range finder was used to detect these features. Montemerlo [7] proposed a different solution to SLAM. He used a rao-blackwellized particle filter that factorized the posterior as a function of the robot path and the Landmarks. Guivant and Nebot [8] introduced another novel method, aimed at reducing the complexity of SLAM. In their algorithm, they used a Compressed Extended Kalman Filter (CEKF), which restricted SLAM iterations to a local area with a small number of landmarks, and only performs a full SLAM when the robot navigates out of the local sub-map and it is necessary to update the full map. FastSLAM and CEKF SLAM used trees as landmarks in their implementation. The process of detecting these landmarks was not trivial since trees could have varying shapes and dimensions (height, diameter, inclination). To account for this variability, Guivant and Nebot [9] began by clustering range readings into semicircles. An average of each cluster represented the expected diameter of the tree associated with that cluster. A Kalman filter was used to track each tree. The diameter of each feature was updated after each scan and then used to evaluate the range and bearing to the center of the trunk.

Although the aforementioned algorithms were able to provide the solution to absolute localization in an unknown environment in real-time, all of them lacked transportability. If, for instance, the robot performing FastSLAM were to move to an area containing no trees, it would fail unless reprogrammed to use features that can be found in the new setting.

This paper will present a methodology that can autonomously specify landmarks for SLAM according to

\[ P(p_t, M | s_t, m_t, n_t), \]  

which states that the probability of having the robot at pose \( p_t \) inside a map \( M \), given the sensor reading \( s_t \), motion \( m_t \), and the mapping \( n_t \) between the observations and the features in the map \( M \). It is evident from (1) that the features must be discernable in order to successfully correlate their observation to the location of the robot. Features that are adequate to conduct SLAM depend on the exteroceptive sensor used and the environment surrounding the robot. Several criteria govern the selection of appropriate features. Firstly, the features must be easily distinguishable by the robot sensor. Secondly, there must be a sufficient number of these landmarks in the mapped environment. Finally, the detection of these landmarks must be fairly insensitive to lighting conditions.

In the following section, the features that have been used throughout the SLAM history will be detailed. Furthermore, the most appropriate features for SmartSLAM will be suggested.

2.1 Historical overview of features used in SLAM

In 1991 Leonard and Durrant-Whyte [1] first referred to the term Simultaneous Localization and Mapping. Experiments were conducted indoors and sonar was used to detect features. Corners, walls, and cylinders were used as features. Faugeras et al. [2] performed localization using a Kalman filter that integrated inertial data with visual cues extracted from a stereo system. In 1997 the PhD thesis by Csorba [3] examined the SLAM problem from a theoretical point of view. In the experimental section of Csorba's thesis, it was assumed that the environment consisted only of point features. These features were the beacons, or landmarks, that were used in SLAM. In 1998 Newman and Durrant-Whyte [4] presented a paper regarding SLAM in an underwater setting. This paper investigated an inertial and terrain based approach to the SLAM problem. The authors performed a ‘focus of attention’ type tracking, in which a particular world frame target is kept in view by the sensor, whilst vehicle motion is occurring. Features were detected by applying a target extraction algorithm to the estimated ocean floor gradient. This estimate was obtained using a high performance terrain tracking sonar system. In 1998 Thrun et al. [5] addressed the issue of building large-scale geometric maps of indoor environments. The landmarks that were used were chosen’s ‘meet-points’, which correspond to intersections, corners, or dead ends in corridors. From 1999 to date, there have been a myriad of papers on SLAM and different approaches of solving it. Listing all these papers would be futile. Rather, only the ones that are the most relevant to this paper will be mentioned. In 2002, DiSana et al. [6] showed that it is possible to remove a great deal of landmarks from the SLAM map without making the map building process statistically inconsistent. The landmarks that the authors used were stationary point targets, such as foreground points, corners and edges. A laser range finder was used to detect these features. Montemerlo [7] proposed a different solution to SLAM. He used a rao-blackwellized particle filter that factorized the posterior as a function of the robot path and the Landmarks. Guivant and Nebot [8] introduced another novel method, aimed at reducing the complexity of SLAM. In their algorithm, they used a Compressed Extended Kalman Filter (CEKF), which restricted SLAM iterations to a local area with a small number of landmarks, and only performs a full SLAM when the robot navigates out of the local sub-map and it is necessary to update the full map. FastSLAM and CEKF SLAM used trees as landmarks in their implementation. The process of detecting these landmarks was not trivial since trees could have varying shapes and dimensions (height, diameter, inclination). To account for this variability, Guivant and Nebot [9] began by clustering range readings into semicircles. An average of each cluster represented the expected diameter of the tree associated with that cluster. A Kalman filter was used to track each tree. The diameter of each feature was updated after each scan and then used to evaluate the range and bearing to the center of the trunk.

Although the aforementioned algorithms were able to provide the solution to absolute localization in an unknown environment in real-time, all of them lacked transportability. If, for instance, the robot performing FastSLAM were to move to an area containing no trees, it would fail unless reprogrammed to use features that can be found in the new setting.

This paper will present a methodology that can autonomously specify landmarks for SLAM according to
the surrounding environment. The robot will be able to specify what features to use according to its entourage. An essential condition for the success of this technique is environment recognition. As such, it is only appropriate to first explain available environment recognition methods before proceeding to an elaborate explanation of our algorithm.

3 Environment classification

Scene classification can be achieved using a bottom-up or a top-down approach. In the former case the environment is inferred from a set of objects that have been recognized in the environment. This is considered a hard problem in the computer vision community and is still unresolved. The order of complexity of such scene recognition algorithms is on par with that of object recognition, which is too costly to be implemented into our algorithm. In the top-down or holistic approach, scenes are recognized by analyzing the spatial organization of structural elements of the images of the scene. The structural composition of each image class provides it with a distinctive signature that is sufficient to discriminate it from other scene classes. Windowed Fourier Transforms, Gabor filters and Wavelet Decomposition filters are the most common frequency decomposition methods that are used to extract image spatial structure. These are holistic approaches, where all pixels in the image contribute to each representation. In this context, Tieu and Viola [10] applied several low-level filters to discover salient features in images. Oliva and Torralba [11] used a Windowed Fourier Transform to acquire a low dimensional representation of the scene, which they termed the Spatial Envelope. Torralba [12] performed Gabor filtering to extract salient features and Torralba et al. [13] applied a steerable pyramid filter to their images to extract feature vectors for the sake of place and object recognition.

The algorithm in this paper on scene recognition is similar to that of Torralba et al. [13] in that it also utilizes a steerable pyramid to extract feature vectors from images. The difference between the algorithms resides in the learning and classification of query images; while the former algorithm uses a Hidden Markov Model (HMM) to learn and classify query scenes, this paper uses a neural network combined with a histogram. The details of the SmartSLAM algorithm including environment recognition and SLAM are presented in the following section.

Figure 1 presents an example of two environment classes with their low-level features extracted via a steerable pyramid. The filtered images were obtained using the code of Simoncelli and Freeman. [14]

4 SmartSLAM

It is the authors’ belief that context determination is a primordial attribute that robots must possess in order to navigate across distinct environments. Before conducting SLAM, the robot dedicates some time to study and thereby categorize the setting around it (i.e. indoor office, outdoor street, outdoor urban, etc.) Once it knows its milieu, the robot picks from a database landmarks that are most available in that category of environment. At this stage of our research only two environment classes were used: Indoor and outdoor. Realistically, the classes will have to be more specific in order to account for all the locations that the robot might visit that exhibit distinctive features for SLAM.

It goes without saying that an exhaustive list of environment classes is not possible, however, the list can be general enough to encompass most of the areas that the robot is likely to visit. Table 1 presents a possible list of environments and corresponding features that could be used to set up a query database. In any case, SmartSLAM will be structured in a modular fashion such that any new environment that the robot might want to visit, which is not in its database, can be added.

SmartSLAM consists of three phases. The first phase deals with environment recognition. A large number of images of various indoor and outdoor urban environments are collected and classified by hand. A steerable pyramid (SP) consisting of six orientations and four scales is then used to filter the collected images, resulting in feature vectors consisting of 24 entries for every pixel. The number of feature vectors is then reduced...
by spatially averaging and sub sampling across each image. The net result is 4 x 4 pixels per image, each having a feature vector consisting of 24 sub-bands. In other words, each image is represented by a feature vector of dimensionality: 24 sub-bands x 16 pixels = 384. The dimensionality of these vectors is further reduced by extracting their first 40 principal components.

Table 1. Possible environment classes and features to conduct SmartSLAM

<table>
<thead>
<tr>
<th>Environment class</th>
<th>Feature used for SLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indoor office-lounge</td>
<td>points</td>
</tr>
<tr>
<td>Indoor corridor</td>
<td>edges</td>
</tr>
<tr>
<td>Outdoor street</td>
<td>lines</td>
</tr>
<tr>
<td>Outdoor park</td>
<td>Trees</td>
</tr>
<tr>
<td>Outdoor urban</td>
<td>Lamp posts</td>
</tr>
<tr>
<td>Outdoor suburban</td>
<td>corners</td>
</tr>
</tbody>
</table>

The second phase of SmartSLAM consists in implementing a machine learning algorithm that is capable of predicting the class of a query image. This phase differs from the work of Torralba et al [13] in that image classification is achieved via a neural network rather than a Hidden Markov Model. Unlike their algorithm, in SmartSLAM a more generic classifier is required that can capture the class (indoor office versus outdoor street) and not location (author1’s office versus author2’s office). Therefore a HMM is not necessary, and adequate classification can be performed by a learning algorithm that is not dependent on heuristics. Furthermore, once the robot recognizes its setting it will be desirable to dedicate all its resources to SLAM, which is in itself a heuristic for mapping and localization. In this context, a neural network (NN) was chosen as the machine learning algorithm for SmartSLAM. This NN is a three-layered 40-40-1 feedforward backpropagation NN with logsig squashing functions, using Levenberg-Marquardt learning with a learning rate of 0.01.

The third and final stage of SmartSLAM represents selection of features that are appropriate for each environment. This phase will be reported in future work.

5 Results

All the SmartSLAM code was written in Matlab and was implemented on a Intel Pentium 4 processor, 3.2 GHz, 1Gbyte RAM workstation. In the preprocessing phase one thousand images of various indoor and outdoor urban environments were collected and labeled according to their environment class. The images were filtered and PCA applied to extract their feature vectors. The computational time for this preprocessing phase amounted to 574s. The feature vectors were then used as training instances for the NN, which trained successfully in 20 epochs and a duration of 540s. Once the training was complete, the NN was tested on 100 images from an indoor setting and 100 images from an outdoor urban setting. Figure 2 reveals sample pictures from the training and test sets.

![Figure 2](image.png)

The NN classified the indoor query images with a success rate of 83% and the outdoor query images with a success rate of 80% (figures 3). A classification was considered ‘correct’ when the error was less than 0.5. The total time required to extract the feature vectors of 100 images and classify them amounted to 61s. This time is expected to be considerably reduced when implemented on the robot, where C language will be used instead of Matlab.

In the second test, images from an outdoor suburban setting were queried using the same network as above. The NN performed poorly and yielded a success rate of 51 percent which is almost equivalent to random guessing. These poor results were expected because the NN was not exposed to any suburban images in its training set. Comparing the suburban images in figure 4 to the urban pictures in figure 2, it is intuitive why the above NN failed. Although both are images of external environments, there is significant difference in the structure of both. In the latter case there are natural objects such as trees and grass that exhibit a distinctive spectral signature.
Furthermore, the suburban setting lacks the tall buildings that are so common in the urban environment. To remedy this setback, the NN had to be retrained by including images of outdoor suburban locations in the training data. This retrained NN yielded 81% correct classification.

In the third and final test, a video sequence of the indoor lab at the American University of Beirut (AUB) (see figure 5) and another sequence of the outdoor AUB campus was tested on the NN. The result was an impressive 95% correct recognition rate for the indoor images and 79% for the outdoor images.

Figures 6 and 7 depict the performance of the NN on these images. It is interesting to note that the variability in the classification error is much lower (i.e. closer to 0) in the indoor than the outdoor environment. The reason for this discrepancy is that an indoor environment is more structured than an outdoor one; a fact that enables the NN to classify indoor settings with more certainty than outdoor ones. In any case, classification results for both environments are acceptable and are indicative of the strong inductive bias of the NN for classification, which permits good generalization.
6 Conclusions and future scope

The contribution of this paper is twofold. Firstly, this is the first algorithm that performs SLAM across multi-environments. SmartSLAM uses environment recognition as a primer for feature selection. The second contribution of this paper is in the area of environment recognition and classification. SmartSLAM classifies images using learning alone. A neural network is trained to classify indoor and outdoor environments. Future work will include a more comprehensive list of environments and investigate a methodology for feature representation and selection.

References


